Electronic Colloquium on Computational Complexity, Report No. 125 (2025)

Hardness Along the Boundary: Towards One-Way Functions from the Worst-case Hardness of Time-Bounded Kolmogorov Complexity

Yanyi Liu*
Cornell Tech
yl2866@cornell.edu

Rafael Pass†
Cornell Tech, Technion and TAU
rafael@cs.cornell.edu

August 20, 2025

Abstract

We consider the question of whether worst-case hardness of the time-bounded Kolmogorov complexity problem, $\mathsf{MINK}^{\mathsf{poly}}$—that is, determining whether a string is time-bounded Kolmogorov random (K^t-random) or not—suffices to imply the existence of one-way functions (OWF).

Roughly speaking, our main result shows that under a natural strengthening of standard-type derandomization assumptions, worst-case hardness of the *boundary* version of this classic problem characterizes OWFs. In more detail, let $\mathsf{boundary\text{-}MINK}^{t_1,t_2}$ denote the problem of, given an instance x, deciding whether (a) $K^{t_2}(x) \geq n-1$, or (b) $K^{t_1}(x) < n-1$ *but* $K^{t_2} > n - \log n$; that is, deciding whether x is K^t-random, or just "near" K^t-random. We say that $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$ if $\mathsf{boundary\text{-}MINK}^{t_1,t_2} \notin \mathsf{ioBPP}$ for all polynomials t_1, t_2.

We show that under a natural strengthening of standard derandomization assumptions (namely, there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$), OWF exist iff $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$. Along the way, we also demonstrate that if we consider the probabilistic version of Kolmogorov complexity (referred to as pK^t) instead, then the characterization holds unconditionally.

We finally observe that for most standard optimization problems, hardness "along boundary" is equivalent to "plain" worst-case hardness, indicating that assuming hardness along the boundary may be WLOG.

*Research partly supported by NSF CNS-2149305.

†Supported in part by NSF Award CNS 2149305, AFOSR Award FA9550-23-1-0387, AFOSR Award FA9550-23-1-0312, AFOSR Award FA9550-24-1-0267, ISF Award 2338/23 and ERC Advanced Grant KolmoCrypt - 101142322. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, the AFOSR, the European Union or the European Research Council Executive Agency.

[1]An extended abstract of this paper appears in *CRYPTO'25*; this is the full version.

ISSN 1433-8092

1 Introduction

A *one-way function* [DH76] (OWF) is a function f that can be efficiently computed (in polynomial time), yet no probabilistic polynomial-time (PPT) algorithm can invert f with inverse polynomial probability for infinitely many input lengths n. Whether one-way functions exist is unequivocally the most important open problem in Cryptography (and arguably the most important open problem in the theory of computation, see e.g., [Lev03]): OWFs are both necessary [IL89] and sufficient for many of the most central cryptographic primitives and protocols (e.g., pseudorandom generators [BM84, HILL99], pseudorandom functions [GGM84], private-key encryption [GM84], digital signatures [Rom90], commitment schemes [Nao91], identification protocols [FS90], coin-flipping protocols [Blu82], and more). These primitives and protocols are often referred to as *private-key primitives*, or "Minicrypt" primitives [Imp95] as they exclude the notable task of public-key encryption [DH76, RSA83]. Additionally, as observed by Impagliazzo [Gur89, Imp95], the existence of a OWF is equivalent to the existence of polynomial-time method for sampling hard *solved* instances for an NP language (i.e., hard instances together with their witnesses).

Complexity-Theoretic Characterizations of OWFs: Kolmogorov Complexity We here focus on the question of whether there exists some "simple" complexity-theoretic characterizations of the existence of OWFs. Recently, [LP20] presented an *average-case* characterization of OWFs through a natural computational problem—the *time-bounded Kolmogorov complexity problem* [Kol68, Sip83, Ko86, Har83]: Let the Kolmogorov complexity of a string x, denoted $K(x)$, be defined as the length of the shortest program that outputs x, and the $t(\cdot)$-bounded version, $K^t(x)$, be defined as the length of the shortest program that outputs the string x within time $t(|x|)$. While determining (or deciding for some particular threshold $s(\cdot)$) the Kolmogorov complexity of a string x is uncomputable, as surveyed by Trakhtenbrot [Tra84], the problem of efficiently computing/deciding K^t-complexity (when t is a polynomial) predates the theory of NP-completeness and was studied in the Soviet Union since the 60s as a candidate for a problem that requires "brute-force search". Indeed, so far no non-trivial attacks are known in the uniform setting (and only very recently a non-trivial attack of circuit size roughly $2^{4n/5}$ was presented in the non-uniform setting [MP23, HIW24]).

The work of [LP20] showed that mild average-case hardness of computing K^t (or simply deciding whether it is large or small) w.r.t. the *uniform* distribution over instances, characterizes the existence of OWF. [LP23b], furthermore, show that if we consider a probabilistic analog of K^t, referred to as *probabilistic* K^t or pK^t [GKLO22]—where for some constant δ (think of $\delta = 2/3$), $pK^t_\delta(x)$ is defined as the smallest ω such that with probability δ over the choice of a random string r, there exists a program of length at most ω that generates x if being provided r—then mild average-case hardness with respect to *any* a-priori polynomially time-bounded sampler over instances also characterizes OWFs (i.e., not just the uniform distribution), and this results also extends to K^t under standard derandomization assumptions.

Does Just *Worst-case* Hardness of K^t Suffice? These above results, however, only apply in the setting of *average-case hardness*. We are here interested in the question of whether simply *worst-case hardness* of computing/deciding K^t suffices to characterize or even just obtain OWFs.

Does worst-case hardness of deciding $K^t(x) > s$ imply the existence of OWFs?

Beyond being interesting in its own right, this question is motivated by the fact that in recent years, there has been great progress (see e.g. [Ila20, ILO20, Ila21, Ila22, LP22b, Hir22, Ila23]) towards showing that the decisional K^t problem is NP-complete; in particular, the very recent work by

Ilango [Ila23] shows NP-completeness in the random oracle model (and thus giving a heuristic NP-completeness reduction). Notably, if OWF can be based on the worst-case hardness of this problem, and the problem can be shown to be NP-complete, then we would base OWF on just the assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

Intriguingly, a result by Hirahara [Hir18] shows that if we were to consider a "gap" version of the K^t problem, where the goal is to distinguish between x such that $K^{t_1}(x) < s$ and $K^{t_2}(x) > s + n^\epsilon$, for all $t_2 = \mathsf{poly}(t_1)$, then worst-case hardness of this problem implies what is referred to as *errorless* average-case hardness (w.r.t. deterministic algorithm).[1] Unfortunately, to apply the result of [LP20], we require the more standard notion of *two-sided* error average-case hardness.

Recently, however, worst-case characterizations of OWF were also obtained [LP23a, HN23], considering some variants of the time-bounded Kolmogorov complexity problem. In particular, [LP23a] characterize OWFs through the problem of determining whether $K^t(x)$ is large or small, but restricting attention to instances x with very large *unbounded* Kolmogorov complexity (i.e., $K(x) > n - O(\log n)$), or alternatively to strings x whose so-called "computational depth", $cd^t(x) = K^t(x) - K(x) < O(\log n)$, is small (whereby "restricting attention" means that we consider worst-case hardness of a promise problem that only considers those instances; that is, any efficient algorithm must fail on one of those instances in the promise). [HN23], in turn, provide a worst-case characterization of a variant of OWFs, referred to as infinitely-often OWF through the problem of "estimating the probability that a random time-bounded program outputs a certain string" (which can be shown to be related to the notion of probabilistic K^t), while restricting attention to instances satisfying an analog of small computational depth (with respect to this complexity notion).[2]

The unappealing aspect of the above characterizations is that once we add the "conditioning", it becomes less clear what the intuitive interpretation of the problems is. On a technical level, the property that we condition on (i.e., computational depth being small, or Kolmogorov complexity being large) is not *decidable*. While it is not hard to modify the condition to become decidable in exponential-time (by considering K^{EXP} instead of K), we still end up with a promise problem where the promise is outside of the polynomial hierarchy, so while "syntactically close" to the original K^t problem, in terms of computational complexity, the problems appear very different.

Our Results in a Nutshell: Boundary Hardness Characterizes OWFs Roughly speaking, the main result of this paper will be that under standard derandomization assumptions, it will suffice to condition on instances having "not too small" *time-bounded* Kolmogorov complexity (as opposed to them having high Kolmogorov complexity). Doing so enables us to obtain the following natural statement: under appropriate derandomization assumptions, worst-case hardness of the problem of deciding whether the time-bounded Kolmogorov complexity of a string x is "very large" or "intermediate" implies, and in fact, characterizes OWFs. We will refer to this as the *boundary time-bounded Kolmogorov complexity problem*. Furthermore, the same characterization holds *unconditionally* once we consider pK^t instead of K^t.

[1] Roughly speaking, *errorless* average-case hardness refers to average-case hardness w.r.t. algorithms that either provide the right answer or $\perp$ (and $\perp$ should only be output with small probability). In other words, hardness holds w.r.t. algorithms that "know" when they fail. This is in contrast to the notion of *two-sided* error hardness where we allow the attacker to fail (without knowing it) on some small set of inputs.

[2] An alternative type of worst-case characterization of OWFs was also obtained in [HN23], where it is shown that OWFs exists iff $\mathsf{NP} \not\subseteq \mathsf{BPP}$ and a certain "distributional"-K^t problem is NP-complete w.r.t. a certain type of (restricted) reductions. Note that simply worst-case hardness of the distributional K^t problem alone is not sufficient to get OWFs Rather the characterization is in terms both a hardness assumption ($\mathsf{NP} \not\subseteq \mathsf{BPP}$) combined with an feasibility assumption (the existence of a certain type of NP-completeness reduction).

1.1 Statement of Results

To state our results more formally, let us first recall the standard $\mathsf{MINK}^{\mathsf{poly}}$ problem; the probabilistic version of the problem, $\mathsf{MINpK}^{\mathsf{poly}}$, is defined analogously (see Section 3.1). For any polynomials $t_1 < t_2$, let MINK^{t_1,t_2} denote the promise problem consisting of:

- YES: $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n-2$.
- NO: $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n-1$,

In essence, the goal is to distinguish between so-called *time-bounded Kolmogorov-random strings* (i.e., x s.t. $K^t(x) \geq n-1$) and those that are not. We say that $\mathsf{MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$ if for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, $\mathsf{MINK}^{t_1,t_2} \notin \mathsf{ioBPP}$.[3]

The *boundary* version (denoted by $\mathsf{boundary\text{-}MINK}^{t_1,t_2}$) is defined as the promise problem consisting of:

- YES: $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n-2$ and $K^{t_2}(x) > n - \log n$.
- NO: $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n-1$.

and $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}}$ is analogously defined. In other words, the problem requires:

> *Deciding whether a string x is* **(a)** *time-bounded Kolmogorov random, or* **(b)** *just "near" time-bounded Kolmogorov random (i.e., having "intermediate" time-bounded Kolmogorov complexity.)*

Let Derand denote the following derandomization assumptions (which is a stronger version of commonly used derandomization assumption: there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$. We are now ready to state our main theorem:

Theorem 1.1. *Assuming* Derand*, OWFs exist if and only if* $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$.

As a warm-up and stepping stone towards this result, we show an *unconditional* characterization of OWF through the worst-case hardness of $\mathsf{boundary\text{-}MINpK}^{\mathsf{poly}}$.

Theorem 1.2. *OWFs exist if and only if* $\mathsf{boundary\text{-}MINpK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$.

A Note on the Derandomization Assumption: Let us provide some context for our new derandomization assumption. The most classic derandomization assumptions in the literature assert the existence of a function $f\colon \{0,1\}^n \to \{0,1\}$ computable in 2^{100n} time that cannot be computed by deterministic [IW97], or non-deterministic [KvM02, MV05, SU05], $2^{0.01n}$-time algorithms with $2^{0.01n}$ advice.[4] (Prior works on the coding theorem—as we shall discuss shortly, this will be a key building block for us—rely on the non-deterministic version (i.e., [GKLO22, LP23b]).)

Our Derand assumption is a natural *strengthening* of the assumption of [KvM02] where, in essence, we require for every k, the existence of a function running in time 2^{100kn} that is secure against attackers with running time $2^{0.01kn}$ (but still only $2^{0.01n}$ advice). In other words, we allow the attacker to run in time $> 2^n$, but the hard function can be computed in time $\gg 2^n$. (Note that, just as for [IW97, KvM02], for each k, there is a $\mathsf{poly}(n)2^{kn}$-time complete hard function.) As for intuition, Derand relies on exactly the same intuition as [IW97, KvM02]: non-determinism and short

[3] Recall that ioBPP denotes the class of promise problems that admit a probabilistic polynomial time algorithm on infinitely many input lengths.

[4] There is nothing special about 100 and 0.01, they could be any constant; we just wrote 100 to make the assumption more concrete, and to enable easier comparing the assumptions with other ones.

advice should not enable polynomial speed-up for computations (and this should hold equally for 2^n [IW97, KvM02] as 2^{kn} time (this paper) computations).

As far as we know this assumption is new, but the idea of considering attackers with running time $> 2^n$ was recently introduced by Chen and Tell [CT21] where a quantitatively stronger version is used: for every $k \geq 1$, exist a function computable in time 2^{100kn} that cannot be computed by 2^{99kn}-time algorithms with $2^{0.99n}$ advice (i.e., they consider an almost-optimal hardness version). On the other hand, [CT21] only make this assumption w.r.t. deterministic attackers, whereas we need it w.r.t. non-deterministic ones. [CT23] considers a non-deterministic version of [CT21], but theirs is again incomparable since they allow the function to be computable by a *non-deterministic* 2^{100kn} algorithm.

1.2 Is Hardness along the Boundary WLOG?

A natural question is whether the $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}}$ problem is equivalent to the $\mathsf{MINK}^{\mathsf{poly}}$ problem (which would resolve the above central question of whether worst-case hardness of $\mathsf{MINK}^{\mathsf{poly}}$ suffices to get OWFs, at least under derandomization assumption). Intuitively, instances on the "boundary" seem to be the hardest to deal with (exactly because they are on the boundary) so one would expect this to be the case. While we cannot prove this formally, we note that for most natural optimization problems, hardness along the boundary is equivalent to simply standard hardness.

For simplicity, consider $3SAT$: Is the "boundary-$3SAT$" problem easier than simply $3SAT$, where in boundary-$3SAT$ you only need to distinguish between (a) fully satisfiable instances, and (b) unsatisfiable instances where we are guaranteed that at least $n - \log n$ clauses can be satisfied (i.e., close to satisfiable clauses)? Indeed, as pointed out to us by Johan Håstad, the answer is no. To solve $3SAT$ using a boundary-$3SAT$ oracle, simply run the decider for the boundary problem; if it answers NO, output NO. Otherwise, iteratively replaces the last (non-dummy) clause with a dummy clause that always is true, and again query the boundary oracle. Note that if we are given a NO instance, this procedure will make sure that we eventually end up in the boundary region and the decider will output NO. And if we are given a YES-instance, the decider will always output YES.

More generally, the above approach works for any optimization problem where we can "smoothly" transform an instance x into an instance x' whose value will eventually increase, but never more than $O(1)$, or even just $O(\log n)$ in any one step. As far as we know, most natural optimization problems indeed satisfy this property and as such their worst-case hardness of their boundary versions is equivalent to worst-case hardness of their standard version.[5] Indeed, this even is the case for the $\mathsf{boundary\text{-}MINK}^{t_1,t_2}$ problem when restricting to the case that $t_1 = t_2$ (at least under appropriate derandomization assumptions). The reason for this is that we can smoothly increase the K^t-complexity of a string x by replacing the last bit of it by a random string[6] and we will finally end up with a string that is required to have K^t-complexity $n - O(\log n)$ (with high probability).

Unfortunately, this argument does not work when we allow $t_2 > t_1$ (as we require for our result). The issue is that there may be a gap between $K^{t_1}(x)$ and $K^{t_2}(x)$ so when we are gradually increasing $K^{t_2}(x)$ to end up in the promise (i.e., in the "boundary region"), we may have accidentally pushed $K^{t_1}(x)$ to become $\geq n - 1$ and inadvertently transformed a YES-instance to a NO-instance!

Perhaps intriguingly, in the average-case characterization of OWF of [LP20] it actually suffices to require hardness of the MINK^{t_1,t_2} for the case that $t_1 = t_2$ (i.e., deciding whether $K^t(x)$ is small or large). In contrast, as we shall discuss in more detail later on, our analysis is non-black box in the attacker, and the parameter t_2 needs to be larger than the running-time of the attacker, which

[5] Most but not all: as pointed out to us by Per Austrin, this may not be the case for the *longest path problem*.

[6] Arguing that this does not *decrease* the K^t complexity non-trivial and requires using a so-called a computational "weak" symmetry-of-information theorem [Ila23], which holds unconditionally for pK^t and also for K^t under appropriate derandomization assumptions [Ila23].

is why we require hardness w.r.t. any $t_2 = \mathsf{poly}(t_1)$; this is similar to the worst-case to errorless average-case reduction of Hirahara [Hir18].

1.3 Proof Overview

We proceed to a proof outline of our results. We focus on explaining how we obtain a OWF from the worst-case hardness of $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}}$ (i.e., the "if" direction of Theorem 1.1). The converse direction will follow mostly using the techniques from [LP23a], which passed through the notion of an "entropy-preserving PRG" constructed in [LP20] (and later improved in [LP23a]).

Towards this, it is instructive to briefly review the worst-case characterization of OWFs in [LP23a] (and how it is proved).

Earlier Approaches and the Key Obstacle As mentioned in the introduction, [LP23a] obtained an initial worst-case characterization of OWFs: they showed that the worst-case hardness of distinguishing whether K^t of an input $x \in \{0,1\}^n$ is at most $n-2$, given that the instance has *time unbounded* Kolmogorov complexity at least $n - \log n$, implies OWFs. (As mentioned before, a related result, relying on similar intuitions, is also proved by [HN23], but for our purposes, the approach in [LP23a] is more relevant.) Roughly speaking, the results of [LP23a] first rely on the result of [LP20] showing that average-case hardness of $\mathsf{MINK}^{\mathsf{poly}}$ implies the existence of OWFs. In more detail, [LP20] presents a reduction for solving $\mathsf{MINK}^{\mathsf{poly}}$ given a OWF inverter (for their specific OWF). [LP23a] showed that this reduction, in fact, also solves the worst-case problem on all instances with high *unbounded* Kolmogorov complexity. This follows from the simple observation that since the average-case reduction fails only with small probability on the uniform distribution, the strings on which it fails must have small (unbounded) Kolmogorov complexity, as they can be enumerated (so we can encode each such string as the index in the enumeration).

We plan to rely on exactly the same reduction. However, to make it work in our setting, we need to argue that if the reduction fails on some instance x, then the *time-bounded* Kolmogorov complexity of x is small—in other words, we require so-called *language compression with efficient decompression* [Sip83, GS85, BLM00, BLM05, Hir21] for the set of instances x on which the reduction fails. In fact, for our purposes, it will suffice to require an efficient decompression algorithm having access to a OWF inverter oracle (as our goal is to establish the existence of OWFs).

Unfortunately, it is an open problem to obtain language compression with efficient decompression for languages in NP (even relative to a OWF inverter). A step towards such language compression was recently taken by Hirahara et al [HIL+23], which shows that a notion of "average-case language compression" can be achieved using a OWF inverter; this, notion, however, will not suffice for us—we require all-input compression.

Our Approach: Language-Specific Instance Compression Rather than generically solving the language compression problem, we will show that for the *specific* language of interest to us (i.e., the set of strings for which the decider fails), we can obtain a short representation of all instances, that can be efficiently decompressed (i.e., their K^{poly}-complexity) is (relatively) small—in more details $K^{\mathsf{poly}}(x) < n - \log n$. This will suffice to show that if the decider fails on some instance x, then $K^{\mathsf{poly}}(x) < n - \log n$, so the instance is not part of the promise of the boundary problem.

On a high-level, we prove the above through the following two steps:

- **Step 1:** Showing that elements on which the decider fails can be sampled with probability $\geq n^c/2^n$ for any c, by an algorithm running in time $T = \mathsf{poly}(n^c)$.

- **Step 2:** Relying on/Proving a so-called "Coding Theorem" [LOZ22] bounding the K^t (or pK^t) complexity of an element by, roughly, the *logarithm* of the inverse of the probability by which it can be sampled.

Taken together, we get that elements on which the decider fails must have K^{poly}-complexity (resp. pK^{poly}-complexity) at most (roughly) $\log(2^n/n^c) = n - c\log n$.

To prove Theorem 1.2 (i.e., the characterization in terms of pK^{poly}), it will suffice to demonstrate Step 2 w.r.t. pK^{poly}, and such a Coding Theorem is already known [LOZ22, San23]. (And as such, the result on pK^{poly} can be viewed as a warm-up case for our result for K^{poly}). In fact, [LOZ22, GKLO22] also provide a Coding Theorem for K^{poly} based on derandomization assumptions, but this Coding Theorem loses $O(\log T)$ in the bound where T is the running time of the sampler, and consequently combined with Step 1 we do not get any non-trivial compressions. As an independent technical contribution, we show how to get a tight Coding Theorem for K^{poly} based on a derandomization assumption; we note, however, that the derandomization assumption that we rely on is a stronger than the one used in [LOZ22, GKLO22] (to prove the non-tight Coding Theorem), but of a similar spirit.

We proceed to provide a more detailed outline of Steps 1 and 2.

Step 1: Recall that our goal is to show that elements on which the [LP20] decider fails can be sampled with probability $\geq n^c/2^n$ for any c, by an algorithm running in time $T = \mathsf{poly}(n^c)$. Towards showing this, we proceed in two steps, letting bad, denote the set of instances on which the reductions from [LP20] fail.

- **Hitting bad through [LP20] Sampling:** In the first step, we show that by the [LP20] construction/reduction, it follows that there exists an efficient way to sample instances in bad. Roughly speaking, the OWF construction samples strings x together with their K^t witness; furthermore, for those instances x we can check whether the reduction fails (since the reduction actually provides a witness when it succeeds).

  Additionally, the one-way function distribution is point-wise multiplicatively close to the uniform distribution (i.e., it dominates the uniform distribution) so each $x \in \{0,1\}^n$ is sampled with probability at least $2^{-n}/O(n)$ by the OWF construction. More precisely,

  - Each string $x \in \{0,1\}^n$ is sampled with probability at least $\frac{1}{O(n)} \cdot 2^{-n}$.
  - The reduction fails (on instance x) only when the inverter fails to invert the [LP20] OWF on x (which thus is efficiently checkable). Consequently, the probability we sample an element on which the reduction fails is ϵ, where ϵ is the failure probability of the OWF inverter.

  It is important to note here that by Yao's hardness amplification theorem [Yao82] (already employed in [LP20]), ϵ can be made into an arbitrarily small polynomial—this point will be crucial to us shortly.

  As a consequence of the above two points, we have that the OWF sampler in this first step samples elements in bad with probability at least $2^{-n}/O(n)$, but the total weight of those elements is ϵ.

- **Boosting the Probabilities through Rejection Sampling:** In the second step, we show how to "boost" the probability of elements in bad. The idea is simple: As the algorithm in step 1 enables checking whether the reduction failed (simply to find the appropriate length K^t witness), we can just repeatedly run the sampler until we find an element on which the

reduction fails. Since the total weight of elements in bad is ϵ, we have that sampling conditioned on ending up in bad, the probability of each element in bad is $2^{-n}/O(n) \times 1/\epsilon$, which can be lowerbounded by $n^c/2^n$ for any choice of c by choosing an appropriately small ϵ (which as mentioned above can be achieved by using Yao's hardness amplification [Yao82]). Of course the issue is that this sampler only runs in expected polynomial time and not strict polynomial time, but by appropriately cutting off its running time we can make sure to not lose too much in the probabilities (essentially by a union bound).

This concludes the argument that elements on which the decider fails can be sampled with high probability.

Step 2: An Improved Coding Theorem for K^{poly} We turn to discussing how all such element (i.e., those on which the decider fails, and consequently by Step 1, that can be sampled with high probability) can be compressed into $n - \log n$ bits (with efficient decompression). As previously discussed, towards this, we will rely on a so-called Coding Theorem. As mentioned above, to prove Theorem 1.2 (i.e., to upperbound pK^{poly}), we can directly combine Step 1 with the known Coding Theorem of [LOZ22, San23], that shows exactly the desired *probabilistic* compression for elements sampled with high probability.

Theorem 1.3 (Coding Theorem for pK^{poly} [LOZ22, San23])**.** *There exists a (universal) constant d such that for any (efficiently) samplable distribution ensemble $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, there exists a polynomial $t(n)$ such that for all sufficiently large $n \in \mathbb{N}$, all $x \in \mathsf{supp}(\mathcal{D}_n)$,*

$$pK^t(x) \leq \lceil \log 1/p_x \rceil + d \log n$$

where $p_x = \Pr[\mathcal{D}_n = x]$.

Additionally, as mentioned, [LOZ22, GKLO22] also provide a Coding Theorem for K^{poly} based on derandomization assumptions, but this Coding Theorem loses $O(\log T)$ in the bound where T is the running time of the sampler; this Coding Theorem will not suffice for us (since whatever gain we get from Step 1 in terms of sampling probability, gets "eaten up" by the overhead in running-time due to the rejection sampling condition on the event bad).

As an independent technical contribution (which we hope may be of independent interest), we show how to get a tight Coding Theorem for K^t based on Derand:

Theorem 1.4 ((Improved) Coding Theorem for K^{poly})**.** *Assuming Derand, there exists a (universal) constant d such that for any (efficiently) samplable distribution ensemble $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, there exists a polynomial $t(n)$ such that for all sufficiently large $n \in \mathbb{N}$, all $x \in \mathsf{supp}(\mathcal{D}_n)$,*

$$K^t(x) \leq \lceil \log 1/p_x \rceil + d \log n$$

where $p_x = \Pr[\mathcal{D}_n = x]$.

We emphasize that compared to the existing Coding Theorem [LOZ22, GKLO22], the above theorem only has a logarithmic overhead in terms of n, whereas in the earlier Coding Theorem, the compression overhead is logarithmic in the runtime of the distribution $\mathcal{D}$. (On the other hand, as mentioned, we are using a slightly stronger derandomization assumption).

Towards proving Theorem 1.4, we will rely on the same approach as [LOZ22]; we rely on the Coding Theorem for pK^{poly} (i.e., Theorem 1.3 above), and next show that, under derandomization assumptions, K^{poly} is no more than pK^{poly} with an additive logarithmic term. Our contribution will be showing a tighter derandomization result (i.e. a tighter correspondence between K^{poly} and pK^{poly}) under a stronger derandomization assumption. We now proceed to show how this is done.

Recall that for any polynomial t, any $x \in \{0,1\}^*$, $pK^t(x)$ is defined to be the smallest ω such that for a 2/3 fraction of the choice of a random string $r \in \{0,1\}^{t(n)}$, there exists a program M of length $\leq \omega$ that produces x given r within time $t(n)$. In order to "derandomize" pK^{poly} into K^{poly}, it suffices to find a succinct description to any such string r, which together with the program M (that exists for r) produces a K^{poly}-witness for x. As observed by [GKLO22], we can obtain such a succinct description of r by relying on a (complexity-theoretic) PRG (with a small seed length) that fools the following non-deterministic circuit $C_{x,\omega}$: $C_{x,\omega}$ takes a random-tape $r \in \{0,1\}^{t(n)}$ as input, guesses a program M of size ω, outputs 1 if $M(r)$ generates x within $t(n)$ steps (and 0 otherwise).[7] We can use a seed of the PRG (that leads $C_{x,\omega}$ to output 1) as the succinct description we need (and thus the seed itself will be our overhead).

Following the *Hardness vs. Randomness paradigm* [NW94], there has been an elegant line of research [KvM02, MV05, SU05] on constructing such PRGs (in particular, those that enable the derandomization of AM to NP). These constructions, however, have seed length that is at least $O(\log t(n))$ when derandomizing circuits of input size $t(n)$ (which is the case for $C_{x,\omega}$). This causes an overhead of $O(\log t(n))$ (as opposed to $d \log n$ for a universal constant d), and indeed, this is why the Coding Theorem of [LOZ22, GKLO22] loses a term of $O(\log t(n))$. We overcome this issue by constructing a new PRG with *near-optimal seed length* (which also may be of independent interest).

(Non-Black-Box) PRGs with Near-Optimal Seed Length Thus, the question is whether we can improve the seed length in these PRG constructions from $O(\log t(n))$ to $d \log n$. Unfortunately, as noted in [SU05], constructions following the standard Hardness vs. Randomness paradigm cannot have a seed length of $o(\log t(n))$! (Roughly speaking, any "black-box" PRG construction from a hard function can be viewed as an extractor [Tre01] and the above bound follows from bounds on the seed length of extractors [NZ96, RTS00].) This barrier was very recently broken using a non-black-box approach in the work of Chen and Tell [CT21], and is closely related to the question of whether "super-efficient derandomization" [DMOZ22] is possible, but their work relies on the existence of OWF and thus is of little help to us.[8]

We here present an alternative approach to overcome this barrier without the use of OWFs. While the seed-length of our PRG will be small, its running-time will still be large, so it will not solve the question of whether super-efficient derandomization is possible without OWFs, but it will suffice for our purposes of derandomizing pK^{poly}.

We proceed to explain our approach. Towards this, let us first recall the standard NW paradigm [NW94]: It provides a PRG construction G that transforms a hard-to-compute function $f \in \{0,1\}^\ell$ (i.e., the truth table of a hard function) into a PRG with output length m. The security reduction shows how to use any attacker distinguishing G^f from random to compute f using $\mathsf{poly}(m)$ bits of (non-uniform) advice.

To get a PRG with small seed length, our idea is to compose two NW-style PRGs: $G(\cdot) = G_1(G_2(\cdot))$ where $G_2 : \{0,1\}^{c_1 \cdot \log n} \to \{0,1\}^{c_2 \cdot \log t(n)}$, $G_1 : \{0,1\}^{c_2 \cdot \log t(n)} \to \{0,1\}^{t(n)}$ (where c_1 and c_2 are universal constants independent of t). (In other words, use one PRG to expand the seed from $O(\log n)$ to $O(\log t(n))$, and next use another PRG on top of this "large" seed.) The key observation is that while the security reduction of the outer PRG G_1 requires $\Omega(t(n))$ bits of advice, the advice will *not* appear in the security game for the inner PRG G_2; in fact, as we shall see, G_2 only needs to handle attackers with $n + O(\log n)$ bits of advice (as opposed to $t(n)$).

To see why this is the case, note that to prove that G is secure in the eyes of $C_{x,\omega}$, we proceed

[7] We remark that $C_{x,\omega}$ has input length $t(n)$ because the "probabilistic" program M runs in time $t(n)$ and can use up to $t(n)$ bits of randomness.

[8] Roughly speaking, a PRG with a small seed enables small simulation overhead when performing derandomization since we have less seed that we need to go over.

by a hybrid argument showing that (a) $G_1(\mathcal{U}_{c_2 \log t(n)})$ and $\mathcal{U}_{t(n)}$ are indistinguishable in the eyes of $C_{x,\omega}(\cdot)$; and that (b) $G_2(\mathcal{U}_{c_1 \log n})$ and $\mathcal{U}_{c_2 \log t(n)}$ are indistinguishable in the eyes of $C_{x,\omega}(G_1(\cdot))$. The existence of G_1 satisfying (a) follows from the standard Hardness vs. Randomness paradigm (under standard derandomization assumptions, and in particular those implied by Derand) since the seed of G_1 is long enough. On the other hand, in (b), G_2 only needs to fool $C_{x,\omega}(G_1(\cdot))$, which has advice size $|x| + |\omega| \leq n + \log n$ and input length $c_2 \log t(n)$; so the seed length only needs to depend logarithmically on n and $\log t(n)$ (which is tiny). The problem, however, is that the runtime of G_1 does contribute to the runtime of the attacker in the security game for G_2 (which leads to an attacker that runs in time polynomial in $t(n)$). This means that the hard function employed in G_2, denoted by f, needs to be hard against algorithms with advice size $O(n, \mathsf{poly} \log t(n))$, but *running time* $\mathsf{poly}(t(n))$, which follows from our derandomization assumption Derand. [9]

A Note on the Non-Black-Box Nature of the Proof: Let us emphasize that the above use of the Coding Theorem makes our security analysis non-black-box in the attacker, and furthermore only works if the attacker is a *uniform* probabilistic polynomial-time algorithm. In more detail, to apply the Coding Theorem (to compress the inputs x on which the decider fails), we require the input x to be samplable in *uniform* polynomial time; as we described above in Step 1, this is shown by applying rejection sampling on the attacker, so if the attacker is non-uniform then we can only sample from the distribution in non-uniform polynomial time and we can no longer apply Step 2.

1.4 Overview of the Paper

In Section 2, we recall some standard definitions; Section 3 contains the formalization of problems we consider and the statement of our main results. Section 4 considers Coding Theorems and additionally abstracts out a more general version of them (including the above rejection sampling step). Section 5 proves the forward direction of the main theorems, and Section 6 proves the backward direction. Finally, in Section 7 we prove the new tighter correspondence between pK^t and K^t (which is used in Section 4); this section also includes the construction of our near-optimal seed-length PRG.

1.5 Open Problems

We briefly note a few open problems.

- **Dealing with non-uniform attackers:** As mentioned above, our results only characterize OWFs secure w.r.t. uniform (i.e., PPT) attackers. While typically security w.r.t. uniform attackers directly yield security w.r.t. non-uniform attacker (one we make non-uniform security assumptions), this does not follow in our setting since our proof is non-black-box in the attacker. We leave open the question of finding a natural problem characterizing OWFs secure also w.r.t. non-uniform PPT attacker. (We note that the problem in [LP23a] works in both the uniform and the non-uniform setting).

- **Basing Security on a Search Problem:** We note that in the average-case setting, the construction of [LP20] can also be based on average-case hardness of the *search* version of K^t complexity problem. Our reduction can also be modified to provide a witness for YES-instances inside the promise (i.e., K^t is "intermediate"), but when K^t is "small" we may no longer efficiently find the witness (we just prove that one exists); if one could prove a

[9] In the literature, a padding trick is typically used to resolve this type of issues (i.e., running f on a larger input length), but this will blow up the seed length by too much for us. To overcome this issue, we here instead rely on a stronger hardness assumption (namely, Derand), in which we decouple the advice size of the attacker from its running time; that is, we assume that there exists $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for *every* k.

Coding Theorem that also enables to efficiently perform the compression (as opposed to just guaranteeing that it exists), then this issue would be resolved and one would have based security on the general (i.e., non-boundary) version of the K^t problem (i.e, $\mathsf{MK}^t\mathsf{P}$).

- **Improving the "width" of the boundary:** Our notion of a boundary problem considers a boundary of width $O(\log n)$. We note that if the width would be improved to $O(n/\log n)$, that would yield a construction of OWFs starting from the worst-case hardness of the standard $\mathsf{MK}^t\mathsf{P}$ problem w.r.t. adversaries running in time $2^{O(n/\log n)}$ (and $t = 2^{O(n/\log n)}$) (i.e., assuming the uniform Perebor conjecture).
- **Relativization barriers:** We mention that while our proof is non-black-box in terms of the attacker, the analysis relativizes and therefore is subject to the barriers in [Imp11, HN22a, HN22b]. We leave open the problem of developing non-relativizing techniques that can go beyond that barrier.

2 Preliminaries

2.1 One-Way Functions

We recall the definition of one-way functions [DH76]. Roughly speaking, a function f is one-way if it is polynomial-time computable, but hard to invert for PPT attackers.

Definition 2.1. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. f is said to be a* one-way function (OWF) *if for every PPT algorithm $\mathcal{A}$, there exists a negligible function μ such that for all $n \in \mathbb{N}$,*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : \mathcal{A}(1^n, y) \in f^{-1}(f(x))] \leq \mu(n)$$

We may also consider a weaker notion of a *weak one-way function* [Yao82], where we only require all PPT attackers to fail with probability noticeably bounded away from 1:

Definition 2.2. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. f is said to be an α-*weak one-way function (α-weak OWF) *if for every PPT algorithm $\mathcal{A}$, for all sufficiently large $n \in N$,*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : \mathcal{A}(1^n, y) \in f^{-1}(f(x))] < 1 - \alpha(n)$$

We say that f is simply a weak one-way function (weak OWF) *if there exists some polynomial $q > 0$ such that f is a $\frac{1}{q(\cdot)}$-weak OWF.*

Yao's hardness amplification theorem [Yao82] shows that any weak OWF can be turned into a (strong) OWF.

Theorem 2.3 ([Yao82])**.** *Assume there exists a weak one-way function. Then there exists a one-way function.*

2.2 Time-bounded Kolmogorov Complexity

We introduce the notion of time-bounded conditional Kolmogorov complexity. Roughly speaking, the *t-time-bounded Kolmogorov complexity*, $K^t(x \mid z)$, of a string $x \in \{0,1\}^*$ conditioned on a string $z \in \{0,1\}^*$ is the length of the shortest program $\Pi = (M, y)$ such that $\Pi(z)$ outputs x in $t(|x|)$ steps.

Formally, fix some universal RAM machine U (with only polynomial overhead), and let $t(\cdot)$ be a running time bound. For any string $x, z \in \{0,1\}^*$, we define

$$K^t(x \mid z) = \min\{w \in \mathbb{N} \mid \exists \Pi \in \{0,1\}^w, U(\Pi(z), 1^{t(|x|)}) = x\}$$

When z is an empty string, we simply denote the quantity by $K^t(x)$. We consider RAM machines (as in [LP22b, GKLO22]) since it allows z to be as long as the running time of the machine Π (or even longer).

Very recently, Goldberg et al [GKLO22] introduced a probabilistic variant of time-bounded Kolmogorov complexity, denoted as pK^t. Let us recall the notion here. Roughly speaking, in the probabilistic version, the program is allowed to be picked after a uniform random string. And a string will have small pK^t-complexity if a short program exists over a large fraction of random strings. We proceed to the formal definition. Let $\delta(n)$ be a probability threshold function. For any string $x \in \{0,1\}^*$, the δ-probabilistic t-bounded Kolmogorov complexity of x [GKLO22], $pK^t_\delta(x)$, is defined to be

$$pK^t_\delta(x) = \min\{w \in \mathbb{N} \mid \Pr[r \leftarrow \{0,1\}^{t(|x|)} : K^t(x \mid r) \leq w] \geq \delta(n)\}$$

The constant δ will be usually considered as being $2/3$. We omit the subscript δ if $\delta = 2/3$.

We refer the reader to Section 3.1 for the definitions of computational problems involving K^{poly} and pK^{poly} we needed in this work.

2.3 Distributions, Random Variables, and Entropy

Let $\mathcal{D}$ be a distribution. We let $\mathsf{supp}(\mathcal{D})$ denote the support of $\mathcal{D}$. We say that $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ is an *ensemble* if for all $n \in \mathbb{N}$, D_n is a probability distribution over $\{0,1\}^n$. We say that an ensemble $\mathcal{D} = \{D_n\}_{n\in\mathbb{N}}$ is (efficiently) *samplable* if there exists a probabilistic polynomial-time Turing machine S such that $S(1^n)$ samples D_n.

For any two random variables X and Y defined over some set $\mathcal{V}$, we let $\mathsf{SD}(X,Y) = \frac{1}{2}\sum_{v\in\mathcal{V}} |\Pr[X = v] - \Pr[Y = v]|$ denote the *statistical distance* between X and Y.

For a random variable X, let $H(X) = \mathsf{E}[\log \frac{1}{\Pr[X=x]}]$ denote the (Shannon) entropy of X, and let $H_\infty(X) = \min_{x\in\mathsf{Supp}(X)} \log \frac{1}{\Pr[X=x]}$ denote the *min-entropy* of X.

The following lemma/fact will be useful for us.

Lemma 2.4 ([LP23a, Lemma 2.6])**.** *Let X be a random variable distributed over $S \subseteq \{0,1\}^n$, E be an set $\subseteq S$. It holds that*

$$\Pr[x \leftarrow X : x \in E] \leq \frac{\log|S| + 1 - H(X)}{\log|S| - \log|E|}$$

3 Main Results

Before presenting the main results in this work, let us formally define the computational problems needed in our results.

3.1 Problem Definitions

The K^{poly} Problem and Its Boundary Version For any polynomials t_1, t_2, $t_1(n) \leq t_2(n)$, define the time-bounded Kolmogorov complexity problem (denoted by MINK^{t_1,t_2}) as the promise problem consisting of

- YES: strings $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n - 2$.

- NO: strings $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n-1$.

In addition, we can define its boundary version (denoted by boundary-MINK^{t_1,t_2}) as

- YES: strings $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n-2$ and $K^{t_2}(x) > n - \log n$.
- NO: strings $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n-1$.

pK^{poly} Problem and Its Boundary Version For any polynomials t_1, t_2, $t_1(n) \leq t_2(n)$, and a parameter $\delta(n) > 0$, define the time-bounded probabilistic Kolmogorov complexity problem (denoted by MINpK$^{t_1,t_2}_\delta$) as the promise problem consisting of

- YES: strings $x \in \{0,1\}^n$, $pK^{t_1}_\delta(x) \leq n - \lceil \log(1/(1-\delta)) \rceil - 2$.
- NO: strings $x \in \{0,1\}^n$, $pK^{t_2}_{1-\delta}(x) \geq n - \lceil \log(1/(1-\delta)) \rceil - 1$.

We remark that the threshold (compared with the threshold in MINK^{t_1,t_2}) is shifted from $n-2$ to $n - \lceil \log(1/(1-\delta)) \rceil - 2$ to ensure that the NO instance set is not empty.

Analogously, we can also define the boundary version (denoted by boundary-MINpK$^{t_1,t_2}_\delta$) as

- YES: strings $x \in \{0,1\}^n$, $pK^{t_1}_\delta(x) \leq n - \lceil \log(1/(1-\delta)) \rceil - 2$ and $pK^{t_2}_{1-\delta}(x) > n - \log n$.
- NO: strings $x \in \{0,1\}^n$, $pK^{t_2}_{1-\delta}(x) \geq n - \lceil \log(1/(1-\delta)) \rceil - 1$.

We will consider $\delta = \frac{2}{3}$ and simply denote MINpK$^{t_1,t_2}_\delta$ (resp boundary-MINpK$^{t_1,t_2}_\delta$) by MINpK^{t_1,t_2} (resp boundary-MINpK^{t_1,t_2}) where $\delta = \frac{2}{3}$.

Finally, for K^{poly} (resp pK^{poly}), we say that MINK$^{\mathsf{poly}} \notin$ ioBPP (resp MINpK$^{\mathsf{poly}} \notin$ ioBPP) if for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, MINK$^{t_1,t_2} \notin$ ioBPP (resp MINpK$^{t_1,t_2} \notin$ ioBPP).

3.2 Main Theorems

We are ready to describe the main results in this work. Our first (and warm-up) result shows that the existence of OWFs is equivalent to the worst-case hardness of the boundary version of MINpK$^{\mathsf{poly}}$.

Theorem 3.1. *OWFs exist if and only if* boundary-MINpK$^{\mathsf{poly}} \notin$ ioBPP*.*

Proof: The theorem follows from (the first bullet of) Theorem 5.1 (which proves the "if" direction), and Theorem 6.2 and Lemma 6.4 (that together prove the "only if" direction). ■

Our main result extends this theorem to obtain a characterization of OWFs based on the worst-case hardness of the boundary version of MINK$^{\mathsf{poly}}$; this time, however, we will rely on a standard derandomization assumption.

Theorem 3.2. *Assume that there exists a constant $\varepsilon > 0$ such that* E $\not\subseteq$ ioNTIME$[2^{kn}]/2^{\varepsilon n}$ *for every $k \in \mathbb{N}$. Then, OWFs exist if and only if* boundary-MINK$^{\mathsf{poly}} \notin$ ioBPP*.*

Proof: The equivalence follows from (the second bullet of) Theorem 5.1 (which proves the "if" direction), and Theorem 6.2 and Lemma 6.3 (that together prove the "only if" direction). ■

4 Source Coding Theorems

In this section, we start by recall the Time-Bounded Coding Theorem for pK^t of [LOZ22] (which in turn is an efficient version of the classic Coding Theorem for unbounded Kolmogorov Complexity). We next present a useful rejection sampling lemma which may be viewed as a generalization of the Coding Theorem.

Finally, we show how to use appropriate PRGs (constructed in Section 7) to also establish a tight Coding Theorem for K^t, and also establish the generalized version. We start by reviewing the Coding Theorem for pK^{poly}, introduced in [LOZ22].[10]

Theorem 4.1 (Source Coding Theorem for pK^{poly} [LOZ22, San23])**.** *For any (efficiently) samplable distribution ensemble $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, there exists a polynomial $t(n)$ such that for all sufficiently large $n \in \mathbb{N}$, all $x \in \mathsf{supp}(\mathcal{D}_n)$,*

$$pK^t(x) \leq \log 1/p_x + 4\log n$$

where $p_x = \Pr[\mathcal{D}_n = x]$.

We turn to proving the following generalized version of the Coding Theorem obtained by relying on rejection sampling.

Lemma 4.2. *For any samplable distribution ensemble $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, any polynomial $q(n)$, there exists a polynomial $t(n)$ such that for any sufficiently large $n \in \mathbb{N}$, if*

$$\Pr[\mathcal{D}_n = \bot] \geq 1 - \frac{1}{q(n)}$$

then, for every $x \in \mathsf{supp}(\mathcal{D}_n)$ such that $x \neq \bot$, it holds that

$$pK^t(x) \leq \log 1/p_x - \log(q(n)/4) + 4\log n$$

where $p_x = \Pr[\mathcal{D}_n = x]$.

Proof: Consider the distribution ensemble $\mathcal{D}' = \{\mathcal{D}'_n\}_{n\in\mathbb{N}}$ sampled by the following algorithm. For each $n \in \mathbb{N}$, $\mathcal{D}'_n$ draws $q(n)/2$ i.i.d. samples $x_1, \ldots, x_{q(n)/2}$ from $\mathcal{D}_n$. $\mathcal{D}'_n$ simply output x_i with the smallest index i for which $x_i \neq \bot$. If all of x_i are $\bot$, $\mathcal{D}'_n$ outputs $\bot$.

We claim that for all sufficiently large n satisfying

$$\Pr[x \leftarrow \mathcal{D}_n : \mathcal{D}_n \neq \bot] \leq \frac{1}{q(n)}$$

and for any $x \in \mathsf{supp}(\mathcal{D}_n)$, $x \neq \bot$, it holds that

$$p'_x \geq p_x \cdot q(n)/4$$

where $p'_x = \Pr[\mathcal{D}'_n = x]$ and $p_x = \Pr[\mathcal{D}_n = x]$. Let δ denote the probability that $\mathcal{D}_n$ does not output

[10] The original statement of this coding theorem has a slightly worse bound but the proof actually proved this stronger statement. This was observed in [San23]

$\perp$. Recall that due to our assumption, $\delta \leq \frac{1}{q(n)}$. It follows from our construction of $\mathcal{D}'_n$ that

$$\begin{aligned}
p'_x &= \sum_{i=1}^{q(n)/2} \Pr_{x_1,\ldots,x_{i-1}\leftarrow\mathcal{D}_n}[x_1 = \perp \wedge \ldots \wedge x_{i-1} = \perp] \cdot \Pr_{x_i\leftarrow\mathcal{D}_n}[x_i = x] \\
&= \sum_{i=1}^{q(n)/2} (1-\delta)^{i-1} \cdot p_x \\
&\geq \sum_{i=1}^{q(n)/2} (1-\delta)^{q(n)/2} \cdot p_x \\
&\geq q(n)/2 \cdot (1 - q(n)\delta/2) \cdot p_x \\
&\geq p_x \cdot q(n)/4
\end{aligned}$$

Finally, notice that the distribution ensemble $\mathcal{D}'$ is efficiently samplable if the ensemble $\mathcal{D}$ is. Thus, by applying Theorem 4.1 to the distribution D' (and letting t be the polynomial guaranteed to exist by Theorem 4.1), it follows that

$$\begin{aligned}
pK^t(x) &\leq \log 1/p'_x + 4\log n \\
&\leq \log 1/p_x - \log(q(n)/4) + 4\log n
\end{aligned}$$

which concludes our proof. ■

We next show how to extend the above to also apply to K^t. We do this by relying on the following tighter connection between pK^t and K^t, the proof of which is postponed to Section 7:

Lemma 4.3 (Derandomizing pK^{poly})**.** *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$. There exists a constant d such that the following holds. For every polynomial t, there exists a polynomial t' such that for every sufficiently long $x \in \{0,1\}^*$,*

$$K^{t'}(x) \leq pK^t(x) + d\log|x|$$

Armed with this wemma, we show that Theorem 4.1 and Lemma 4.2 also hold for K^{poly} under the assumption Derand.

Theorem 4.4 (Source Coding Theorem for K^{poly})**.** *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$. There exists a constant d such that the following holds.*

For any (efficiently) sampleable distribution ensemble $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, there exists a polynomial $t(n)$ such that for all sufficiently large $n \in \mathbb{N}$, all $x \in \mathsf{supp}(\mathcal{D}_n)$,

$$K^t(x) \leq \log 1/p_x + (d+4)\log n$$

where $p_x = \Pr[\mathcal{D}_n = x]$.

Proof: This theorem directly follows from Lemma 4.3 and Theorem 4.1. ■

Lemma 4.5. *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$. There exists a constant d such that the following holds.*

For any samplable distribution ensemble $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, any polynomial $q(n)$, there exists a polynomial $t(n)$ such that for any sufficiently large $n \in \mathbb{N}$, if

$$\Pr[\mathcal{D}_n = \perp] \geq 1 - \frac{1}{q(n)}$$

then, for every $x \in \mathsf{supp}(\mathcal{D}_n)$ *such that* $x \neq \perp$, *it holds that*

$$K^t(x) \leq \log 1/p_x - \log(q(n)/4) + (d+4)\log n$$

where $p_x = \Pr[\mathcal{D}_n = x]$.

Proof: This Lemma follows from the proof of Lemma 4.2 by replacing the use of Theorem 4.1 by Theorem 4.4. ■

5 OWFs from Hardness of boundary-MINpK$^{\mathsf{poly}}$

We turn to proving the forward direction of our main theorems.

Theorem 5.1. *The following statements hold:*

- *If for all polynomials* t_1, t_2, $t_1(n) \leq t_2(n)$,

  $$\text{boundary-MINpK}^{t_1,t_2} \notin \mathsf{ioBPP}$$

  Then, OWFs exist.

- *Assume that* $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\Omega(n)}$ *for every* $k \in \mathbb{N}$. *If for all polynomials* t_1, t_2, $t_1(n) \leq t_2(n)$,

  $$\text{boundary-MINK}^{t_1,t_2} \notin \mathsf{ioBPP}$$

  Then, OWFs exist.

Proof: The first bullet follows by Lemma 5.2 and Theorem 5.3 (proven below), and Theorem 2.3. The second bullet follows by Lemma 5.2 and Theorem 5.4 (proven below), and Theorem 2.3. ■

In fact, we will prove a slightly more general result from which the above theorem follows. Towards this, we will introduce new promise problems regarding pK^{poly} (and K^{poly}).

For any polynomials $t_1(n), t_2(n), t_1(n) \leq t_2(n)$, any threshold function $s(n) > 0$, and any constant $\beta > 0$, define the promise problem $\Pi^{t_1,t_2}_{pK,\beta}[s]$ (for pK^{poly}) as

- YES instances: $x \in \{0,1\}^*$, $pK^{t_1}_{2/3}(x) \leq s(|x|)$, $pK^{t_2}_{1/3}(x) > pK^{t_1}_{2/3}(x) - \beta \log |x|$.
- NO instances: $x \in \{0,1\}^*$, $pK^{t_1}_{1/3}(x) > s(|x|)$.

We will also define $\Pi^{t_1,t_2}_{K,\beta}[s]$ for K^{poly}, by replacing pK with K in the above definition (and simply ignoring the threshold for pK). Roughly speaking, the problem $\Pi^{t_1,t_2}_{K,\beta}[s]$ (resp $\Pi^{t_1,t_2}_{pK,\beta}[s]$) generalizes boundary-MINK^{t_1,t_2} (resp boundary-MINpK^{t_1,t_2}) in the following ways: (1) it considers an arbitrary threshold function s (as opposed to being fixed to $n-2$); (2) the "boundary" could be potentially wider ($\beta \log |x|$ against $\log |x|$); (3) the (Kolmogorov complexity) time bound for NO instances is t_1 (as opposed to t_2) (which allows the promise problem to consider more instances).

Thus, as it is a generalization of the boundary problem, the hardness of boundary-MINpK^{t_1,t_2} will imply the hardness of $\Pi^{t_1,t_2}_{pK,\beta}[s]$ for $\beta = 1$ and $s(n) = n - \lceil \log 3 \rceil - 2$. (The same statement (but replacing $s(n)$ with $n-2$) also holds for K^{poly}.)

Lemma 5.2. *For any* $t_1(n), t_2(n), t_1(n) \leq t_2(n)$. *Let threshold* $s(n) = n - \lceil \log 3 \rceil - 2$, $\beta = 1$,

$$\text{boundary-MINpK}^{t_1,t_2} \notin \mathsf{ioBPP} \Rightarrow \Pi^{t_1,t_2}_{pK,\beta}[s] \notin \mathsf{ioBPP}$$

In addition, the same statement also holds when pK *is replaced by* K *and* $s(n)$ *by* $s(n) = n - 2$.

Proof: Let Π_{YES} (resp Π_{NO}) denote the set of YES instances (resp NO instances) of the promise problem $\Pi^{t_1,t_2}_{pK,\beta}[s]$. We first show that $\mathsf{boundary\text{-}MINpK}^{t_1,t_2}_{\mathsf{YES}} \subseteq \Pi_{\mathsf{YES}}$. For any $x \in \mathsf{boundary\text{-}MINpK}^{t_1,t_2}_{\mathsf{YES}}$, it holds that

$$pK^{t_1}_{2/3}(x) \leq n - \lceil \log 3 \rceil - 2 = s(n),\ pK^{t_2}_{1/3}(x) > s(n) - \log n \geq pK^{t}_{2/3}(x) - \beta \log n$$

Thus, $x \in \Pi_{\mathsf{YES}}$. In addition, we claim that $\mathsf{boundary\text{-}MINpK}^{t_1,t_2}_{\mathsf{NO}} \subseteq \Pi_{\mathsf{NO}}$. This follows from the fact that for any $x \in \mathsf{boundary\text{-}MINpK}^{t_1,t_2}_{\mathsf{NO}}$,

$$pK^{t_1}_{1/3}(x) \geq pK^{t_2}_{1/3}(x) > n - \lceil \log 3 \rceil - 2 = s(n)$$

since $t_1(n) \leq t_2(n)$. Finally, we remark that all the above still holds if we replace pK with K (and $s(n)$ by $s(n) = n - 2$). ■

We turn to proving that $\Pi^{t_1,t_2}_{pK,\beta}[s] \notin \mathsf{ioBPP}$ implies OWFs.

Theorem 5.3. *Assume that there exist a polynomial $t_1(n)$, a threshold $0 < s(n) < n - 1$, and a constant $\beta > 0$ such that for all sufficiently large polynomials $t_2(n)$, $\Pi^{t_1,t_2}_{pK,\beta}[s] \notin \mathsf{ioBPP}$. Then, weak one-way functions exist.*

Proof: Consider any polynomial $t_1(n)$, any threshold $s(n)$, and any constant β. We consider the function $f : \{0,1\}^{\lceil \log(n) \rceil + n + t_1(n)} \to \{0,1\}^*$, which takes an input $\ell || \Pi' || r$ where $|\ell| = \lceil \log(n) \rceil$, $|\Pi'| = n$ and $|r| = t_1(n)$, outputs

$$f(\ell || \Pi' || r) = \ell || U(\Pi(r), 1^{t_1(n)}) || r$$

where Π is a prefix of Π' and Π is of length ℓ (where the bit-string ℓ is interpreted as an integer $\in [n]$).

This function is only defined over some input lengths, but by an easy padding trick, it can be transformed into a function f' defined over all input lengths, such that if f is weakly one-way (over the restricted input lengths), then f' will be weakly one-way (over all input lengths): $f'(x')$ simply truncates its input x' (as little as possible) so that the (truncated) input x now becomes of length $n' = \lceil \log(n) \rceil + n + t_1(n)$ for some n and outputs $f(x)$. This will decrease the input length by a polynomial factor (since t_1 is a polynomial) so the padding trick can be applied here.

We now show that f is a weak OWF (over the restricted input length). Let $q(n) = 90n^{5+2\beta}$. We assume for contradiction that f is not $\frac{1}{q}$-weak one-way. (In the proof below, although the input length of f we consider is $m = \lceil \log(n) \rceil + n + t_1(n)$ for some n, we will view n as the "security parameter" and analyze the one-wayness of f on input length $m = m(n)$ (but computing the running time and the inversion probability in terms of the security parameter n). Since n and m are polynomially related, we can still conclude that f is weak one-way.) Then, there exists a PPT attacker $\mathcal{A}$ that inverts f with probability at least $1 - \frac{1}{q(n)}$ for infinitely many n.

We will consider the distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$, which helps us compress the instances on which $\mathcal{A}$ fails to invert f. $\mathcal{D}$ proceeds as follows: for each $n \in \mathbb{N}$, runs the function f on uniformly sampled input $\leftarrow \mathcal{U}_{\lceil \log(n) \rceil + n + t_1(n)}$. Denote the output of f by (ℓ, y, r). Next, the distribution feeds the output of f to the attacker $\mathcal{A}$ and checks whether $\mathcal{A}$ succeeds in inverting (ℓ, y, r). If $\mathcal{A}$ succeeds, it simply outputs $\perp$. If $\mathcal{A}$ does not succeed, it will output the string y.

Notice that $\mathcal{D}$ will mostly output $\perp$, and will output $\perp$ with probability $\geq 1 - \frac{1}{q(n)}$ on "security parameter" n for which $\mathcal{A}$ succeeds in inverting f. In addition, the ensemble $\mathcal{D}$ can be efficiently sampled (since both f and $\mathcal{A}$ are efficient). Let t_2 be the polynomial we obtain from Lemma 4.2 when plugging in the distribution $\mathcal{D}$ and the polynomial q.

We turn to constructing a PPT algorithm M to decide the promise problem $\Pi^{t_1,t_2}_{pK,\beta}[s]$. Our algorithm M, on input z, samples a random string $r \in \{0,1\}^{t_1(n)}$. Then, it runs $\mathcal{A}(i||z||r)$ for every $i \in [n]$ where i is represented as a $\lceil \log(n) \rceil$-bit string, and outputs 1 if and only if the length of the shortest program Π output by $\mathcal{A}$, which on input r produces the string z within $t_1(n)$ steps, is at most $s(n)$. Since $\mathcal{A}$ runs in polynomial time, our algorithm will also terminate in polynomial time.

We next show that M is a worst-case algorithm (that succeeds infinitely often) for $\Pi^{t_1,t_2}_{pK,\beta}[s]$ (which will be a contradiction and concludes the proof). Fix any sufficiently large n on which $\mathcal{A}$ succeeds in inverting f.

We first show that M will output 1 with probability at least 3/5 on any YES instance of $\Pi^{t_1,t_2}_{pK,\beta}[s]$. Assume for contradiction that M on input some YES instance of $\Pi^{t_1,t_2}_{pK,\beta}[s]$, z, outputs 1 with probability $< 3/5$. Notice that since z is a YES instance, we have that

$$pK^{t_1}_{2/3}(z) \leq s(n) \text{ and } pK^{t_2}_{1/3}(z) > pK^{t_1}_{2/3}(z) - \beta \log n \tag{1}$$

Let $w = pK^{t_1}_{2/3}(z)$. It follows that for at least a 2/3 fraction of randomness $r \in \{0,1\}^{t_1(n)}$, the length of the shortest program that on input r produces the string z, w_r, will be at most w. We refer to such r as being "good", and let G denote the set of good randomness for z. Consider the set $S = \{(w_r, z, r) : r \in G\}$. Since M outputs 1 with probability $< 3/5$, $\mathcal{A}$ must fail to invert f with probability at least

$$\frac{2}{3} - \frac{3}{5} \geq \frac{1}{15}$$

on uniformly random element $\in S$. However, each element in S will be sampled with probability at least

$$\frac{1}{n} \cdot \frac{1}{2^{w_r}} \cdot \frac{1}{2^{|r|}} \geq \frac{1}{n} 2^{-w-t_1(n)}$$

in the one-way function experiment. Thus, the probability that distribution $\mathcal{D}_n$ samples the string z, p_z, will be at least

$$\frac{1}{n} 2^{-w-t(n)} \cdot |S| \cdot \frac{1}{15} \geq \frac{2}{45n} 2^{-w}$$

Now we can apply Lemma 4.2 to conclude that the $pK^{t_2}_{1/3}$-complexity of z is at most

$$\begin{aligned} pK^{t_2}_{1/3}(z) \leq pK^{t_2}_{2/3}(z) &\leq \log 1/p_z - \log(q(n)/4) + 4 \log n \\ &\leq w - \log(q(n)/(90n^5)) \\ &= pK^{t_1}_{2/3}(z) - 2\beta \log n \end{aligned}$$

which contradicts to Equation 1.

It remains to show that if $z \in \{0,1\}^n$ is a NO instance, then $M(z)$ will output 1 with probability $< 2/5$. Notice that since z is a NO instance, $pK^{t_1}_{1/3} > s(n)$. Therefore, there exists no more than 1/3 fraction of random tapes r for which there is a program of length $\leq s(n)$ that taking r as input produces z. Finally, note that M will output 1 only when it finds such a program. It follows that $M(z)$ outputs 1 with probability at most $1/3 < 2/5$. ∎

Finally, we show that the implication also holds if we consider K^{poly} (instead of pK^{poly}). This is not a direct consequence of Theorem 5.3 and Lemma 4.3 (which would prove a gap version of the result). Nevertheless, it essentially follows from similar ideas, and we here provide a proof.

Theorem 5.4. *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$.*

If there exist a polynomial $t_1(n)$, a threshold $0 < s(n) < n-1$, and a constant $\beta > 0$ such that for all sufficiently large polynomials $t_2(n)$, $\Pi^{t_1,t_2}_{K,\beta}[s] \notin \mathsf{ioBPP}$, then, weak one-way functions exist.

Proof: Consider the function $f : \{0,1\}^{n+\lceil \log(n) \rceil} \to \{0,1\}^*$, which given an input $\ell || \Pi'$ where $|\ell| = \lceil \log(n) \rceil$ and $|\Pi'| = n$, outputs

$$\ell || U(\Pi, 1^{t_1(n)})$$

where Π is the ℓ-bit prefix of Π'. Note that U only has polynomial overhead, so f can be computed in polynomial time.

This function is only defined over some input lengths, but by an easy padding trick, it can be transformed into a function f' defined over all input lengths, such that if f is weak one-way (over the restricted input lengths), then f' will be weak one-way (over all input lengths): $f'(x')$ simply truncates its input x' (as little as possible) so that the (truncated) input x now becomes of length $n' = n + \lceil \log(n) \rceil$ for some n and outputs $f(x)$.

Assume for contradiction that f is not $\frac{1}{q(n)}$-weak one-way where $q(n) = 12n^{5+2\beta+d}$ (where d is the universal constant in Lemma 4.5). Then, there exists PPT attacker $\mathcal{A}$ such that the attacker $\mathcal{A}$ inverts the function f with probability at least $1 - \frac{1}{q(n)}$ for infinitely many n. We will consider the distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ (that helps us compress instances on which the inverter $\mathcal{A}$ fails): For each $n \in \mathbb{N}$, $\mathcal{D}_n$ runs the function f on uniformly sampled input $\leftarrow \mathcal{U}_{n+\lceil \log(n) \rceil}$. Denote the output of f by (ℓ, y). Next, $\mathcal{D}_n$ feeds the output of f to the attacker $\mathcal{A}$ and checks whether $\mathcal{A}$ succeeds in inverting (ℓ, y). If $\mathcal{A}$ succeeds, $\mathcal{D}_n$ simply outputs $\perp$. If $\mathcal{A}$ does not succeed, $\mathcal{D}_n$ will output the string y.

Notice that $\mathcal{D}_n$ will mostly output $\perp$, and will not output $\perp$ with probability $\leq \frac{1}{q(n)}$ on input length n for which $\mathcal{A}$ succeeds in inverting f. In addition, the ensemble $\mathcal{D}$ can be efficiently sampled (since both f and $\mathcal{A}$ are efficient). Let t_2 be the polynomial we obtain from Lemma 4.5 when plugging in the distribution $\mathcal{D}$ and the polynomial q.

We turn to constructing a PPT algorithm M to decide $\Pi^{t_1,t_2}_{K,\beta}[s]$ infinitely often. Our algorithm M, on input z, runs $\mathcal{A}(i||z)$ for every $i \in [n]$ where i is represented as a $\lceil \log(n) \rceil$-bit string, and outputs 1 if and only if the length of the shortest program Π output by $\mathcal{A}$, which produces the string z within $t_1(n)$ steps, is at most $s(n)$. Since $\mathcal{A}$ runs in polynomial time, our algorithm will also terminate in polynomial time. Fix some sufficiently large n on which the inverter $\mathcal{A}$ succeeds in inverting f.

We next show that our algorithm will output 1 with probability at least $2/3$ on input YES instances of $\Pi^{t_1,t_2}_{K,\beta}[s]$ of length n. Assume for contradiction that M outputs 1 with probability $< 2/3$ on some YES instances $z \in \{0,1\}^n$. We have that

$$K^{t_1}(z) \leq s(n) \text{ and } K^{t_2}(z) > K^{t_1}(z) - \beta \log n \tag{2}$$

Let $w = K^{t_1}(z)$. Note that there must exist a program of size w such that the program outputs z in time $t_1(n)$. Therefore, (w, z) will be sampled with probability

$$\frac{1}{n} 2^{-w}$$

in the one-way function experiment. In addition, since $\Pr[M(z) = 1] < 2/3$, $\mathcal{A}$ must fail to invert f on input $(w||z)$ with probability at least $1/3$. So, the probability that $\mathcal{D}_n$ samples the string z, p_z, is at least

$$\frac{1}{3n} 2^{-w}$$

It follows from Lemma 4.5 that the K^{t_2} complexity of z is at most

$$\begin{aligned}
&\log 1/p_z - \log(q(n)/4) + (d+4)\log n \\
&\leq w + \log(3n) - \log(q(n)/4) + (d+4)\log n \\
&= K^{t_1}(z) - \log(q(n)/(12n^{d+5})) \\
&= K^{t_1}(z) - 2\beta \log n
\end{aligned}$$

which contradicts to Equation 2.

We finally prove that if z is a NO instance (and thus $K^{t_1}(z) > s(n)$), $M(z)$ will never output 1. Note that $M(z)$ will output 1 only when it finds a K^{t_1}-witness of z with length no more than $s(n)$, and there is no such witness if $K^{t_1}(z) > s(n)$, which proves our claim and concludes the proof. ■

6 Hardness of boundary-MINK$^{\mathsf{poly}}$ from OWFs

We here prove the worst-case hardness of boundary-MINK$^{\mathsf{poly}}$ from the existence of OWFs. The proof is very similar in nature to the proof from [LP20] (with some strengthenings from [LP23a]).

We start by recalling the notion of a conditionally-secure entropy-preserving pseudorandom generator [LP20] (and its strengthened version in [LP23a]), which will be an important tool in our proof.

Definition 6.1. *An efficiently computable function $g : \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}$ is a $\varepsilon(\cdot)$-*conditionally-secure α-entropy-preserving pseudorandom generator (ε-cond α-EP-PRG) *if there exist a sequence of events $= \{E_n\}_{n\in\mathbb{N}}$ such that the following conditions hold:*

- **(pseudorandomness):** *For every* PPT *attacker $\mathcal{A}$ and sufficiently large $n \in \mathbb{N}$,*

$$|\Pr[s \leftarrow \{0,1\}^n : \mathcal{A}(1^n, g(\mathcal{U}_n)) = 1 \mid E_n] - \Pr[r \leftarrow \{0,1\}^{m(n)} : \mathcal{A}(1^n, \mathcal{U}_m) = 1]| < \varepsilon(n), \quad (3)$$

- **(entropy-preserving):** *For all sufficiently large $n \in \mathbb{N}$, $H([g(\mathcal{U}_n \mid E_n)]_n) \geq n - \alpha \log n$ (where $[x]_n$ denotes the n-bit prefix of the string x).*

We refer to the constant α as the entropy-loss constant.

We say that $g : \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}$ has rate-1 efficiency if its running time on input length n is at most $m + O(m^\varepsilon)$ for some constant $\varepsilon < 1$. Recall that a rate-1 efficient cond EP-PRG can be constructed from OWFs [LP20, LP23a].

Theorem 6.2 (Cond EP-PRG from OWFs [LP21, LP23a])**.** *Assume that OWFs exist. Then, for any constant $\alpha > 0, \gamma > 0$, there exists a rate-1 efficient 0.1-cond α-EP-PRG $g : \{0,1\}^n \to \{0,1\}^{n+\gamma \log n}$.*

We remark that in [LP20], they only obtain a α-cond EP-PRG for some fix constant α; [LP23a] improves the construction to show that this result holds for *every* constant $\alpha > 0$.

We proceed to showing that cond EP-PRG implies the worst case hardness of boundary-MINK$^{\mathsf{poly}}$.

Lemma 6.3. *Assume that there exists a rate-1 efficient ε-cond α-EP-PRG $g : \{0,1\}^n \to \{0,1\}^{n+\gamma \log n}$ for $\varepsilon = \alpha = \gamma = 0.1$. Then, for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$,* boundary-MINK$^{t_1,t_2} \notin$ ioBPP.

Proof: Consider any polynomial t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$. Let $m(n) = n + \gamma \log n$ denote the output length of g, and let $\{E_n\}$ denote the sequence of events associated with g.

Suppose for contradiction that boundary-MINK$^{t_1,t_2} \in$ ioBPP. It follows that there exists a PPT algorithm M such that M decides boundary-MINK^{t_1,t_2} infinitely often. (In addition, we can assume without loss of generality that M succeeds with probability ≥ 0.99.) Since $m(n+1) - m(n) \leq 1 + \gamma \leq 2$, there exists a constant $0 \leq b \leq 1$ such that M succeeds on infinitely many m of the form $m = m(n) - b$. We will consider the function g' defined the same as g but truncating the last bit if $b = 1$. Note that g' is a also cond EP-PRG (trivially, since g is). For the ease of presentation, we will present the proof for the case where $b = 0$ (and g' is the same as g). (The proof below can be naturally adapted to the case of $b = 1$.)

We proceed to constructing a distinguisher $\mathcal{A}$ (using M) to break the cond EP-PRG $g : \{0,1\}^n \to \{0,1\}^{m(n)}$. On input 1^n and a string $x \in \{0,1\}^m$, our distinguisher $\mathcal{A}$ simply outputs $M(x)$. M runs in polynomial time, so $\mathcal{A}$ is a PPT algorithm. Fix some sufficiently large n and $m = m(n)$. The following two claims will show that $\mathcal{A}$ succeeds in breaking g on input length n with advantage 2ε, and thus a contradiction.

Claim 1. *$\mathcal{A}(1^n, \mathcal{U}_m)$ will output 0 with probability at least $\frac{1}{2} - 0.01$.*

Proof: Notice that (by a standard counting argument) a random string $r \in \{0,1\}^m$ will have K^{t_2}-complexity at least $m-1$. In addition, since M succeeds on input length m, $M(r)$ outputs 0 with probability at least 0.99 if $K^{t_2}(r) \geq m-1$. Then, the claim follows from a Union bound. ■

Claim 2. *$\mathcal{A}(1^n, g(\mathcal{U}_n \mid E_n))$ will output 0 with probability at most $0.2 + 0.01$.*

Proof: Since $g : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a rate-1 efficient cond EP-PRG with seed length n, we have that for any $y \in \{0,1\}^n$,

$$K^{t_1}(g(y)) \leq n + O(1) \leq n + \gamma \log n - 2 = m - 2$$

(by considering the program hardwiring the code of g (of $O(1)$ bits) and the seed y (of n bits), which runs g on y and produces $g(y)$ in time $t_2(m)$.) Let

$$X = g(\mathcal{U}_n \mid E_n), \ S = \mathsf{supp}(X)$$

denote the random variable for the pseudorandom string (conditioned on E_n) and its support set. Since g is entropy preserving, it holds that $H(X) \geq n - \alpha \log n$. In addition, observe that $2^{H(X)} \leq |S| \leq 2^n$. Let

$$Z = \{x \in S : K^{t_2}(x) \leq m - \log m\}$$

be the set of strings $\in S$ that violates the promise of $\mathsf{boundary\text{-}MINK}^{t_1,t_2}$. By a standard counting argument, $|Z| \leq 2^{m - \log m + 1}$. Notice that for any $x \in S$ but $x \notin Z$, $M(x)$ will output 1 with probability ≥ 0.99. In addition, using Lemma 2.4, we can show that $X \in Z$ with probability at most

$$\begin{aligned}
&\frac{\log |S| + 1 - H(X)}{\log |S| - \log |Z|} \\
\leq &\frac{n + 1 - (n - \alpha \log n)}{n - \alpha \log n - (m - \log m + 1)} \\
= &\frac{0.1 \log n + 1}{\log m - \alpha \log n - \gamma \log n - 1} \\
\leq &\frac{0.1 \log n + 1}{0.8 \log n - 1} \\
\leq &\ 0.2
\end{aligned}$$

Thus, $M(X)$ outputs 0 with probability at most

$$\begin{aligned}
\Pr[M(X) = 0] = &\Pr[X \in Z] \Pr[M(X) = 0 \mid X \in Z] + \Pr[X \notin Z] \Pr[M(X) = 0 \mid X \notin Z] \\
\leq &\Pr[X \in Z] + \Pr[M(X) = 0 \mid X \notin Z] \\
\leq &0.2 + 0.01
\end{aligned}$$

which concludes the claim. ■

■

We remark that the same implication also holds with respect to pK^{poly}.

Lemma 6.4. *Assume that there exists a rate-1 efficient ε-cond α-EP-PRG $g : \{0,1\}^n \to \{0,1\}^{n+\gamma \log n}$ for $\varepsilon = \alpha = \gamma = 0.1$. Then, for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$,* $\mathsf{boundary\text{-}MINpK}^{t_1,t_2} \notin \mathsf{ioBPP}$.

Proof: The proof of Lemma 6.3 essentially also proves this lemma, we here simply clarify the changes that have to be made when adapting the proof of Lemma 6.3 to proving this lemma. Let $\mathcal{A}$ be the distinguisher constructed in the proof of Lemma 6.3. It suffices to prove the following two claims.

Claim 3. *$\mathcal{A}(1^n, \mathcal{U}_m)$ will output 0 with probability at least $\frac{1}{2} - 0.01$.*

Proof: By the probabilistic incompressibility lemma (c.f. [GKLO22, Lemma 20]), it follows that with probability at least $\frac{1}{2}$, a random string $r \in \{0,1\}^m$ will have $pK^{t_2}_{1/3}$-probability at least $m - \lceil \log 1/(1/3) \rceil - 2$. In addition, $M(r)$ will output 0 with probability ≥ 0.99 if $pK^{t_2}_{1/3}(r) \geq m - \lceil \log 3 \rceil - 2$. ■

Claim 4. *$\mathcal{A}(1^n, g(\mathcal{U}_n \mid E_n))$ will output 0 with probability at most $0.2 + 0.01$.*

Proof: Since $g : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a rate-1 efficient cond EP-PRG with seed length n, we have that for any $y \in \{0,1\}^n$,

$$pK^{t_1}_{2/3}(g(y)) \leq K^{t_1}(g(y)) \leq m - 2$$

(where $K^{t_1}(g(y))$ was bounded in the proof of Lemma 6.3). Define X, S in the same way as in Claim 2. Define

$$Z = \{x \in S : pK^{t_2}_{1/3}(x) \leq m - \log m\}$$

with respect to pK-complexity. Again, by the probabilistic incompressibility lemma (c.f. [GKLO22, Lemma 20]), it holds that $|Z| \leq 2^{m - \log m + 1} \cdot 3$. Therefore, the probability that $X \in Z$ is at most

$$\frac{\log |S| + 1 - H(X)}{\log |S| - \log |Z|} \leq \frac{0.1 \log n + 1}{0.8 \log n - 3} \leq 0.2$$

And the claim will follow from the (rest of the) proof of Claim 2. ■

■

7 Derandomizing pK^{poly} and Seed-Optimal PRGs

In this section, provide a proof of Lemma 4.3; that is, we show how to upperbound K^{poly} by pK^{poly} plus an additive logarithmic term, assuming derandomization assumptions. As described in the introduction, we achieve this result by presenting a new construction of a PRG against non-deterministic programs with near-optimal seed length.

7.1 Complexity-theoretic PRGs and Non-uniform Programs

Towards derandomizing pK^{poly}, let us recall the notion of a complexity-theoretic PRGs.

Definition 7.1 (Complexity-theoretic PRG)**.** *Let $g : \{0,1\}^{\ell(m)} \to \{0,1\}^m$ be a function computable in polynomial time in its output length. We say that g is a* complexity $\varepsilon(\cdot)$-pseudorandom generator (complexity ε-PRG) against complexity class $\mathcal{C}$ *if for any distinguisher $D \in \mathcal{C}$, all $m \in \mathbb{N}$,*

$$|\Pr[D(g(\mathcal{U}_{\ell(m)})) = 1] - \Pr[D(\mathcal{U}_m) = 1]| < \varepsilon(m)$$

We refer to ε as distinguishing gap*. We simply say that g is a* complexity-theoretic PRG *if g is a ε-PRG for $\varepsilon = \frac{1}{6}$.*

We will consider complexity-theoretic PRGs that fool non-adaptive non-uniform D-oracle programs, which we introduce here.

Definition 7.2 ((Non-adaptive) Non-uniform Programs)**.** *A non-adaptive non-uniform D-oracle program A on n-bit inputs consists of a pair of non-uniform programs A_{pre} and A_{post}. The program A_{pre} on input $x \in \{0,1\}^n$ outputs queries $q_1, \ldots, q_k$. The program A_{post} receives x together with $a_1, \ldots, a_k$ (where $a_i = 1$ if and only if $q_i \in D$), and it outputs a single bit.*

The running time (resp advice complexity) of A is defined to be the sum of the running time (resp advice complexity) of A_{pre} and A_{post}. We refer to k as the query complexity of A.

7.2 Constructing Complexity-Theoretic PRG with Seed Length $O(\log n)$

Our start point is the "standard" complexity-theoretic PRG that can be used to derandomize AM.

Theorem 7.3 ([SU05])**.** *Assume that $\mathsf{E} \not\subseteq \mathsf{ioNSIZE}[2^{\Omega(n)}]$, there exists a complexity-theoretic PRG $g : \{0,1\}^{O(\log n)} \to \{0,1\}^n$ against $\mathsf{NSIZE}[O(n)]$.*

However, the above PRG falls short in its seed lengt: if we use it to directly derandomize pK^t, it incurs an $O(\log t(|x|))$ overhead. Towards building the PRG we need, we first call the following result from [SU06], which enables us to transform nondeterministic hardness to SAT-oracle hardness.

Theorem 7.4 ([SU06] (c.f. [CT23, Theorem 5.11]))**.** *There exists a universal constant c_{RC} such that the following holds. For any integer $k \geq 1$, constant $0 < \varepsilon < 1/200$, if $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{2kn+2\varepsilon c_{\mathsf{RC}} n}]/2^{2\varepsilon n}$, then E does not have an (infinitely-often) non-adaptive non-uniform SAT-oracle program with running time 2^{kn}, advice complexity $2^{\varepsilon n}$, and query complexity $2^{\varepsilon n}$.*

Proof: This Theorem is obtained by taking $\varepsilon' = 1/2, k' = 2^{2\varepsilon n}, T = 2^{2kn+2\varepsilon c_{\mathsf{RC}} n}, \alpha = 2^{2\varepsilon n}$ into [CT23, Theorem 5.11] (and due to collisions in the notations, we denote k (resp ε) in [CT23, Theorem 5.11] by k' (resp ε')). Also notice that $E \notin \mathsf{ioNTIME}[T]/k'$ implies that E does not have single-value nondeterministic program with advice complexity k' and running time T. ■

We rely on the notion of *black-box PRG construction from a worst-case hard function f* [STV01, Vad12]. Roughly speaking, this notion of black-box PRG from a function f requires the existence of an efficient oracle algorithm that given (a) some fixed advice string, and (b) black-box access to any distinguisher for the PRG, is able to compute function f. We remark that we here only consider deterministic reductions R that reconstruct each bit of f with probability 1 (whereas [Vad12] considered probabilistic reductions). Notice that probabilistic reductions can be made to be deterministic at the price of requiring a longer advice string.

Definition 7.5 (Reconstructive PRG (c.f. [Vad12, Definition 7.65]))**.** *Let $g : 1^n \times 1^m \times \{0,1\}^d \to \{0,1\}^m$ be a (deterministic) oracle algorithm, and let $k(\cdot)$ be functions. We say that g is a k-*reconstructive PRG construction *if there exist a (deterministic) algorithm R such that for every $f : [n] \to \{0,1\}$ and $T : \{0,1\}^m \to \{0,1\}$, if*

$$|\Pr[T(g^f(1^n, 1^m, \mathcal{U}_d)) = 1] - \Pr[T(\mathcal{U}_m) = 1]| \geq \frac{1}{6}$$

then there exists a string $z \in \{0,1\}^{k(n,m)}$ such that for all $i \in [n]$,

$$R^T(z, i) = f(i)$$

We next observe that the Sudan-Trevisan-Vadhan PRG [STV01] obtain by combining a locally list-decodable error correcting code [STV01] and the Nisan-Wigderson PRG construction [NW94] yields a strongly black-box construction of a PRG (as argued in [Vad12]).

Theorem 7.6 ([STV01]; see also [Vad12, Theorem 7.67], [LP22a, Theorem 3.11])**.** *There exists a k-reconstructive PRG construction $g : 1^n \times 1^m \times \{0,1\}^d \to \{0,1\}^m$ such that for every $m \in \mathbb{N}$, $n \geq m$, $f : [n] \to \{0,1\}$ the following conditions are satisfied:*

1. *Explicitness: g^f is computable in uniform time $\mathsf{poly}(m, n)$.*
2. *Seed length: $d(n, m) = O(\log^2 n / \log m)$.*
3. *Reduction running time: $t(n, m) = \mathsf{poly}(m, \log n)$.*
4. *Reduction advice complexity: $k(n, m) \leq t(n, m)$.*

In addition, the reduction R only makes non-adaptive queries.

Now we are ready to construct the complexity-theoretic PRG we need.

Theorem 7.7. *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$. There exists an constant c such that the following holds. For every polynomial t, there exists a complexity-theoretic PRG $G : \{0,1\}^{c \log n} \to \{0,1\}^n$ against $\mathsf{NTIME}[t(n)]/n$.*

Proof: Notice that we start with the hardness of E against non-uniform nondeterministic programs, whereas to use the reconstructive PRG construction, we need to rely on hardness against non-adaptive non-uniform SAT-oracle programs. Thus, we first apply Theorem 7.4 to transform hardness. Let $\varepsilon' = \min\{\varepsilon, 1/200\}$. And it follows that for every $k \geq 1$, E does not have an (infinitely often) non-adaptive non-uniform SAT-oracle program with running time 2^{kn}, advice complexity and query complexity $2^{\varepsilon' \cdot n}$.

Given the hardness assumption, we will be needing a (standard) padding argument (that works for every $k \geq 1$). Let $\theta \leq \varepsilon'/c_0$ be a sufficiently small constant (where c_0 is a universal constant that will be fixed later). We will use (the truth table of) a hard function $\in \mathsf{E}$ on input $\log n$ to construct a PRG with output length $m = n^\theta$.

Consider any polynomial $t(m)$. We proceed to building our complexity-theoretic PRG secure against $\mathsf{NTIME}[t(m)]/m$. Let k be a (large enough) integer such that $m^{k/4} \geq t(m)$. By the argument above, there exists a language $L_k \in \mathsf{E}$ such that L_k does not have an (infinitely often) non-adaptive non-uniform SAT-oracle programs with running time 2^{kn}, advice complexity and query complexity $2^{\varepsilon' \cdot n}$. Let $f_{k,\ell} \in \{0,1\}^{2^\ell}$ denote the truth table of L_k on input length ℓ. For any output length $m \in \mathbb{N}$, let $n = m^{1/\theta}$, and our PRG G_k is defined as

$$G_k \stackrel{\text{def}}{=} g^{f_{k,\log n}}(1^n, 1^m) : \{0,1\}^d \to \{0,1\}^m$$

where g is the reconstructive PRG construction in Theorem 7.6.

Notice that $g^{f_{k,\log n}}$ can be computed in time polynomial in m since both the truth table $f_{k,\log n}$ and the oracle machine g have running time $\mathsf{poly}(n, m) = \mathsf{poly}(m)$. In addition, the seed length $d = d(n, m) = O(\log^2 n / \log m) = O((1/\theta)^2 \log m)$. Let c be a constant such that $d(n, m) \leq c \log m$ and we conclude that G_k has seed length $c \log m$.

We turn to arguing the security of G_k. Assume (for contradiction) that for some $m \in \mathbb{N}$, there exists a nondeterministic non-uniform program $D \in \mathsf{NTIME}[t(m)]/m$ such that

$$|\Pr[D(G_k(\mathcal{U}_d)) = 1] - \Pr[D(\mathcal{U}_m) = 1]| \geq \frac{1}{6}$$

Let R be the deterministic reduction for g as in Theorem 7.6, and let $t_R(n, m)$ be the running time of R. Note that $t_R(n, m) = \mathsf{poly}(m, \log n) \leq \mathsf{poly}(m)$ and R also has advice complexity $t_R(n, m)$. By

the reconstructive property of g, it follows that there exists an advice string $z_R \in \{0,1\}^{t_R(n,m)}$ such that for all $i \in [n]$,

$$R^D(z_R, i) = f(i)$$

Therefore, R^D with advice z_R decides the hard language L_k on input length $\ell = \log n$.

In order to reach a contradiction, it remains to argue that $R^D(z_R, \cdot)$ can be implemented by a non-adaptive non-uniform SAT-oracle program. Recall that D is a nondeterministic non-uniform program with running time $t(m)$ and advice complexity m. Let z_D denote the advice string that D receives. Consider the following two programs $A_{\mathsf{pre}}, A_{\mathsf{post}}$ that act as follows.

- A_{pre} receives z_R and z_D as advice. On input $y \in [n]$, it uses z_R to simulate the reduction R on input y and obtains the queries $x_1, \ldots, x_{t_R}$ that R makes to D. (Note that since R is also non-adaptive, we can obtain those queries.) For each x_i, notice that $D(x_i)$ can be transformed to a SAT formula ϕ_i of length $t(m) \log t(m)$ such that $\phi_i \in \mathsf{SAT}$ iff $D(x_i) = 1$ in time $t(m)^2$ (using the advice z_D for D). Finally, A_{pre} outputs $\phi_1, \ldots, \phi_{t_R}$.
- A_{pre} receives z_R as advice. On input $y \in [n]$ together with $a_1, \ldots, a_{t_R}$, it simply simulates $R^D(z_R, y)$ but replacing the i-th answer obtained from D with a_i, and outputs what the reduction outputs.

Notice that the attacker $A = (A_{\mathsf{pre}}, A_{\mathsf{post}})$ decides L_k on input length ℓ, and it has advice complexity

$$2|z_R| + |z_D| \leq 2t_R(n,m) + m \leq \mathsf{poly}(m, \log n) \leq n^{O(1)\theta} = 2^{O(1)\theta \cdot \ell}$$

running time

$$t_R(n,m) \cdot t(m)^2 + t_R(n,m) \leq 2^{O(1)\theta \cdot \ell} \cdot t(m)^2 \leq 2^{O(1)\theta \cdot \ell} m^{k/2} \leq 2^{k \cdot \ell}$$

and query complexity

$$t_R(n,m) \leq 2^{O(1)\theta \cdot \ell}$$

By picking $\theta < \varepsilon'/c_0$ for some sufficiently large constant c_0, we have that both the advice complexity and the query complexity are $\leq 2^{\varepsilon' \cdot \ell}$. ∎

7.3 Derandomizing pK^{poly}

We are now ready to use the above-constructed PRG to derandomize pK^t.

Lemma 7.8 (Derandomizing pK^{poly}, Lemma 4.3 restated)**.** *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$. There exists a constant d such that the following holds. For every polynomial t, there exists a polynomial t' such that for every sufficiently long $x \in \{0,1\}^*$,*

$$K^{t'}(x) \leq pK^t(x) + d \log |x|$$

Proof: Consider any polynomial t, and some sufficiently long string $x \in \{0,1\}^n$. Let w denote the pK^t-complexity of x. Let g be the complexity-theoretic PRG guaranteed to exist in Theorem 7.3 (since that the assumption we make in this Lemma trivially implies the assumption needed in Theorem 7.3). Also let c be the constant in Theorem 7.7.

We start with the nondeterministic non-uniform program D_1 defined as follows. D_1 takes a "random tape" $r \in \{0,1\}^{t(n)}$ as input, and guesses a pK^t-witness Π of x (with respect to the random tape r) of length no more than w. Finally, D_1 outputs 1 if Π on input r produces the string x. (Let $T(n)$ be the polynomial that bounds the running time of D_1.) Notice that the running time of D_1,

$T = T(n)$, is at most $O(t(n) \log t(n))$, and the advice complexity of D_1, ℓ, is $|x| + |w|$. It follows from Theorem 7.3 that D_1 cannot distinguish the output of g on a seed of length $O(\log T)$ from the uniform distribution with advantage $1/6$.

However, the seed length of g, $O(\log T)$, is still too long, and we need to apply another complexity-theoretic PRG to further reduce the seed length. We next consider the nondeterministic non-uniform program D_2, which on input $\in \{0,1\}^\ell$, uses its first $O(\log T)$ bits (denote the string by v), and computes $r = g(v)$. Finally, D_2 outputs 1 if $D_1(r)$ outputs 1. Observe that D_2 runs in time $T(n) \leq T(\ell)$ and uses $\leq \ell$ bits of advice. Let G be the complexity-theoretic PRG for the polynomial T guaranteed to exists by Theorem 7.7. It follows that D_2 cannot distinguish the output of G on seed length $c \log \ell$ from the uniform distribution with advantage $1/6$.

Lastly, by the above argument, it holds that

$$\begin{aligned}\Pr[D_2(G(\mathcal{U}_{c\log \ell})) = 1] &\geq \Pr[D_2(\mathcal{U}_\ell) = 1] - \frac{1}{6} \\ &= \Pr[D_1(g(\mathcal{U}_{O(\log T)})) = 1] - \frac{1}{6} \geq \Pr[D_1(\mathcal{U}_{t(n)}) = 1] - \frac{1}{3}\end{aligned}$$

In addition, it follows from the definition of D_1 (together with $pK^t(x) = w$) that D_1 will output 1 with probability at least $2/3$ on the uniform distribution. Thus,

$$\Pr[D_2(G(\mathcal{U}_{c\log \ell})) = 1] \geq \frac{1}{3}$$

Now we are ready to show that x also has small $K^{t'}$-complexity for some polynomial t'. Let s be a seed of length $c \log \ell$ such that $D_2(G(s)) = 1$. Let r be the random tape obtained by running g on the first $O(\log T)$-bit of $G(s)$. Since $D_2(G(s)) = 1$, there exists a program Π such that Π on input r will output x in time $t(n)$. Consider the program M with the polynomial t, code of g and G, the integer n, the seed s (of length $\leq 2c \log n$), and the program Π (of length $\leq w$) hardcoded. M uses s to obtain the string r and output the string obtained by running Π on r. Notice that the PRG G, g, and the program Π all run in polynomial time in n, and let $t'(n)$ be the polynomial such that M runs in time $t'(n)$. In addition, the program M has description length

$$\log n + 2 \log n + 4c \log n + w \leq pK^t(x) + d \log n$$

for some constant d (that does not depend on t). Thus, we conclude that

$$K^{t'}(x) \leq pK^t(x) + d \log n$$

■

8 Acknowledgements

We are very grateful to Johan Håstad, Per Austrin and Noam Mazor for helpful conversations. In particular, we are grateful to Johan Håstad for pointing out that the border version of SAT (and 3SAT) is equivalent to the standard (i.e., non-border) version (and with his permission, we have included his proofs of these observations in the paper). Thanks so much!

References

[BLM00] Harry Buhrman, Sophie Laplante, and Peter Bro Miltersen. New bounds for the language compression problem. In *Proceedings 15th Annual IEEE Conference on Computational Complexity*, pages 126–130. IEEE, 2000.

[BLM05] Harry Buhrman, Troy Lee, and Dieter van Melkebeek. Language compression and pseudorandom generators. *computational complexity*, 14(3):228–255, 2005.

[Blu82] Manuel Blum. Coin flipping by telephone - A protocol for solving impossible problems. In *COMPCON'82, Digest of Papers, Twenty-Fourth IEEE Computer Society International Conference, San Francisco, California, USA, February 22-25, 1982*, pages 133–137. IEEE Computer Society, 1982.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[CT21] Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 283–291, 2021.

[CT23] Lijie Chen and Roei Tell. When arthur has neither random coins nor time to spare: Superfast derandomization of proof systems. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 60–69, 2023.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DMOZ22] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. *Journal of the ACM*, 69(6):1–55, 2022.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC '90*, pages 416–426, 1990.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. In *FOCS*, 1984.

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C Oliveira. Probabilistic kolmogorov complexity with applications to average-case complexity. In *37th Computational Complexity Conference (CCC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GS85] Andrew Goldberg and Michael Sipser. Compression and ranking. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 440–448, 1985.

[Gur89] Yuri Gurevich. The challenger-solver game: variations on the theme of p=np. In *Logic in Computer Science Column, The Bulletin of EATCS*. 1989.

[Har83] J. Hartmanis. Generalized kolmogorov complexity and the structure of feasible computations. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 439–445, Nov 1983.

[HIL+23] Shuichi Hirahara, Rahul Ilango, Zhenjian Lu, Mikito Nanashima, and Igor C Oliveira. A duality between one-way functions and average-case symmetry of information. *Cryptology ePrint Archive*, 2023.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 247–258, 2018.

[Hir21] Shuichi Hirahara. Average-case hardness of np from exponential worst-case hardness assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 292–302, 2021.

[Hir22] Shuichi Hirahara. Np-hardness of learning programs and partial mcsp. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 968–979. IEEE, 2022.

[HIW24] Shuichi Hirahara, Rahul Ilango, and Ryan Williams. Beating brute force for compression problems. *Electronic Colloquium on Computational Complexity*, 2024. https://eccc.weizmann.ac.il/report/2023/171/.

[HN22a] Shuichi Hirahara and Mikito Nanashima. Finding errorless pessiland in error-prone heuristica. In *37th Computational Complexity Conference (CCC 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.

[HN22b] Shuichi Hirahara and Mikito Nanashima. On worst-case learning in relativized heuristica. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 751–758. IEEE, 2022.

[HN23] Shuichi Hirahara and Mikito Nanashima. Learning in pessiland via inductive inference. 2023.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989.

[Ila20] Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and AC^0[p]. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, pages 34:1–34:26, 2020.

[Ila21] Rahul Ilango. The minimum formula size problem is (eth) hard. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 427–432. IEEE, 2021.

[Ila22] Rahul Ilango. Constant depth formula and partial function versions of mcsp are hard. *SIAM Journal on Computing*, (0):FOCS20–317, 2022.

[Ila23] Rahul Ilango. Sat reduces to the minimum circuit size problem with a random oracle. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 733–742, 2023.

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference, CCC 2020*, pages 22:1–22:36, 2020.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory '95*, pages 134–147, 1995.

[Imp11] Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for np. In *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 104–114. IEEE, 2011.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if e requires exponential circuits: Derandomizing the xor lemma. In *STOC '97*, pages 220–229, 1997.

[Ko86] Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(3):9–33, 1986.

[Kol68] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.

[KvM02] Adam R Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[Lev03] L. A. Levin. The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003.

[LOZ22] Zhenjian Lu, Igor C Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded kolmogorov complexity. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020.

[LP21] Yanyi Liu and Rafael Pass. Cryptography from sublinear time hardness of time-bounded kolmogorov complexity. In *STOC*, 2021.

[LP22a] Yanyi Liu and Rafael Pass. Leakage-resilient hardness vs randomness. *Electronic Colloquium on Computational Complexity*, 2022. https://eccc.weizmann.ac.il/report/2022/113/.

[LP22b] Yanyi Liu and Rafael Pass. On one-way functions from np-complete problems. In *Proceedings of the 37th Computational Complexity Conference*, pages 1–24, 2022.

[LP23a] Yanyi Liu and Rafael Pass. On one-way functions and the worst-case hardness of time-bounded kolmogorov complexity. *Cryptology ePrint Archive*, page 1086, 2023.

[LP23b] Yanyi Liu and Rafael Pass. One-way functions and the hardness of (probabilistic) time-bounded kolmogorov complexity w.r.t. samplable distributions. In *CRYPTO'23*, 2023.

[MP23] Noam Mazor and Rafael Pass. The non-uniform perebor conjecture for time-bounded kolmogorov complexity is false. *Cryptology ePrint Archive*, 2023.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.

[RSA83] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[San23] Rahul Santhanam. An algorithmic approach to uniform lower bounds. In *38th Computational Complexity Conference (CCC 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 330–335. ACM, 1983.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM (JACM)*, 52(2):172–216, 2005.

[SU06] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *computational complexity*, 15(4):298–341, 2006.

[Tra84] Boris A Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[Vad12] Salil P Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il